UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/760,129	01/12/2001	Miguel Miranda	IMEC193.001AUS	2250

20995 7590 07/15/2004

KNOBBE MARTENS OLSON & BEAR LLP
2040 MAIN STREET
FOURTEENTH FLOOR
IRVINE, CA 92614

EXAMINER	
FOWLKES, ANDRE R	

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 07/15/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

[illegible]

Office Action Summary	**Application No.** 09/760,129	**Applicant(s)** MIRANDA ET AL.
	Examiner Andre R. Fowlkes	**Art Unit** 2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *06 May 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-37* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-37* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed on 5/6/04.

2. The objection to the information disclosure statement is withdrawn, in view of applicants amendment.

3. The objections to the specification are withdrawn, in view of applicants amendment.

4. The objections to the claims are withdrawn, in view of applicants amendment.

5. Applicant's arguments with respect to claims 1 and 35 have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-5, 11, 12, 16, 19-23, 30-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Blickstein, U.S. Patent No. 5,577,253 in view of Balasa et al. (Balasa), "Transformation of Nested Loops with Modulo Indexing to Affine Recurrences," Parallelization Techniques for Uniform Algorithms, World Scientific Pub., 1994, p. 1-12.

As per claim 1, Blickstein discloses **a method of optimizing address expressions within source-level code, wherein the source-level code describes the functionality of an application to be executed on a digital device** (col. 1 lines 40-41, "This invention relates to compilers (that optimize address expressions in source code for execution on) digital computers")**, the method comprising:**

- inputting a first source-level code that describes the functionality of the application, the first source-level code comprising address computation code (col. 3 lines 10-12, "Each front end scans and parses the source modules (i.e. the first source level code comprising address computation code), and generates, from them, an intermediate language representation of the program expressed in the source code."), **and a plurality of arrays with address expressions** (col. 34 lines 8-9, "The first operand … is an address expression representing the base address of the array")

- transforming the first source-level code into a second source-level code that describes substantially the same functionality as the first source-level code (col. 3 lines 10-15, "Each front end scans and parses the source modules (i.e. the first source level code), and generates, from them, an intermediate language representation of the program expressed in the source code (i.e. a second source level code). This intermediate representation is constructed to represent (the same functionality as the first source level code for)"),

- wherein the second source-level code has fewer costly **operations than the first source-level code** (col. 18 line 55 – col. 19 line 59, "the definition and detection of induction variables and inductive expressions will be discussed … this is a straight forward Do loop, I being the loop control variable. Notice that the inductive expression I*4 increases by 4 on each trip through the loop. By introducing a new variable I2, we can replace the multiplication with an addition, which is a less expensive operation ... This optimization is known as induction variable elimination"),

- **wherein the digital device comprises at least one processor** (col. 2 line 1, "a processor").

Blickstein doesn't explicitly disclose that **the address computation code or one of the address expressions has nonlinear operations**

However, Balasa, in an analogous environment, discloses that **the address expression is nonlinear** (p.2 lines 5-7, "The most important practical subclass of the non-linear extensions consist of modulo expressions of affine indexing functions").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Balasa into the system of Blickstein to have induction analysis applied to at least one address expression which **is nonlinear**. The modification would have been obvious because one of ordinary skill in the art would want to utilize the teachings of Balasa to optimize nonlinear expressions because they are a very important class of expressions for the multi-dimensional signal and data processing field (Balasa, p. 2, lines 5-14).

As per claim 2, the rejection of claim 1 is incorporated and further, Blickstein doesn't explicitly disclose that **the nonlinear operations are selected from the group**

comprising: modulo operations, integer division operations, power operations and multiplication.

However, Balasa, in an analogous environment, discloses that **the nonlinear operations are selected from the group comprising: modulo operations, integer division operations, power operations and multiplication** (p.2 lines 5-7, "The most important practical subclass of the non-linear extensions consist of modulo expressions of affine indexing functions").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Balasa into the system of Blickstein to have the nonlinear operations are selected from the group comprising: **modulo operations, integer division operations, power operations and multiplication**. The modification would have been obvious because one of ordinary skill in the art would want to utilize the teachings of Balasa to optimize nonlinear expressions because they are a very important class of expressions for the multi-dimensional signal and data processing field (Balasa, p. 2, lines 5-14).

As per claim 3, the rejection of claim 1 is incorporated and further, Blickstein discloses **a first code transforming, a second code transforming and a third code transforming, the first code transforming comprising algebraic code transformations and common subexpression elimination code transformations, the second code transforming including code hoisting and the third code transforming including induction analysis** (col. 2 lines 30-31, "various optimizing techniques (ordered transformations) are used", and col. 3 lines 49-52, "This mechanism is used by the global optimizer to determine legal and effective optimizations, including common subexpression expression recognition and code motions", and col. 4 lines 4-5, "In addition to finding induction variables, this optimization finds inductive expressions").

As per claim 4, the rejection of claim 3 is incorporated and further, Blickstein discloses that **first the first code transforming is executed, thereafter the second code transforming and thereafter the third code transforming** (col. 2 lines 30-31, "various optimizing techniques (ordered transformations) are used").

As per claim 5, the rejection of claim 1 is incorporated and further, Blickstein discloses that the **arrays are multi-dimensional arrays** (col. 34 line 27, "multidimensional arrays").

As per claim 11, the rejection of claim 1 is incorporated and further, Blickstein discloses **the method is independent of the digital device architecture** (col. 3 lines 3-4, "the generic back end provides the functions of optimization").

As per claim 12, the rejection of claim 11 is incorporated and further, Blickstein discloses that **the method does not use detailed knowledge of the target device of**

the source code(col. 3 lines 3-4, "the generic back end provides the functions of optimization").

As per claim 16, the rejection of claim 1 is incorporated and further, Blickstein discloses that **transforming comprises applying an algebraic transformation on at least one of the address expressions or part of the address calculation code, wherein the algebraic transformation replaces at least a part of the address expressions or part of an expression within the address computation code by another equivalent expression** (col. 3 lines 49-52, "This mechanism is used by the global optimizer to determine legal and effective optimizations, including common subexpression expression (elimination)).

As per claim 19, the rejection of claim 1 is incorporated and further, Blickstein discloses that **the transformation comprises common subexpression elimination on at least one of the address expressions or part of the address computation code** (col. 13 lines 36-37, "address arithmetic")**, and wherein the common subexpression elimination detects a subexpression that is found within at least two expressions within either the address expressions or the address computation code and replaces the subexpression with a single variable, wherein the variable is computed via the subexpression higher in the code, the variable computation becoming part of the address computation code** (col. 12 line 66 – col. 13 line 3, "As an essential step in detecting common subexpressions (CSEs), invariant expressions, and opportunities for code motion, the optimizer in the back end must be able to determine when two expression tuples are guaranteed to compute the same value").

As per claim 20, the rejection of claim 19 is incorporated and further, Blickstein discloses that **the common subexpression elimination decreases the amount of nonlinear operations within the source-level code** (col. 12 line 66 - col. 13 lines 3, "As an essential step in detecting common subexpressions (CSEs) ... the optimizer in the back end must be able to determine when two expression tuples are guaranteed to compute the same value", and col. 4 lines 5-7, "this optimization finds inductive expressions (nonlinear operations), which are expressions that can be computed as linear functions").

As per claim 21 the rejection of claim 1 is incorporated and further, Blickstein discloses that **at least one piece of address calculation code is located within a selected scope and transforming comprises code hoisting wherein a part of the piece of address calculation code is moved outside the scope** (col. 12 line 66 - col. 13 lines 3, "As an essential step in detecting ... opportunities for code motion (code hoisting), the optimizer in the back end must be able to determine when two expression (address calculation code) tuples are guaranteed to compute the same value").

As per claim 22, the rejection of claim 1 is incorporated and further Blickstein discloses that **the moving of the part of the piece of address calculation code reduces the amount of executions of nonlinear operations within the part of the piece of address calculation code** (col. 12 line 66 - col. 13 lines 3, "As an essential step in detecting ... opportunities for code motion, the optimizer in the back end must be able to determine when two expression tuples are guaranteed to compute the same value", and col. 4 lines 5-7, "this optimization finds inductive expressions, which are (nonlinear) expressions that can be computed as linear functions").

As per claim 23, the rejection of claim 21 is incorporated and further Blickstein discloses that **code hoisting is applied across the scope of a loop construct** (col. 3 lines 49-52, "This mechanism is used by the global optimizer to determine legal and effective optimizations, including ... code motions").

As per claim 30, the rejection of claim 1 is incorporated and further, Blickstein discloses that **modulo related algebraic transformation combined with a common subexpression elimination and code hoisting are executed iteratively** (col. 2 lines 30-31, "various optimization techniques are ... implemented ... Commonly-used optimizations are code motions, strength reduction (modulo related, and other algebraic transformations), etc", and col. 188 lines 4-10, "a first code optimization and to perform a second code optimization ... said code optimization performed using ... common sub expression elimination (and) strength reduction").

As per claim 31, the rejection of claim 1 is incorporated and further Blickstein discloses that an **induction analysis based step is applied on at least one address expression, the induction analysis step replaces the address expression with single pointer arithmetic and a single conditional** (col. 18 line 55 – col. 19 line 59, "the definition and detection of induction variables and inductive expressions will be discussed ... this is a straight forward Do loop, I being the loop control variable. Notice that the inductive expression I*4 increases by 4 on each trip through the loop. By introducing a new variable I2, we can replace the multiplication with an addition, which is a less expensive operation ... This optimization is known as induction variable elimination").

Blickstein doesn't explicitly disclose that the **address expression is piece wise linear.**

However, Balasa, in an analogous environment, discloses that the **address expression is piece wise linear** (p. 2 lines 10-19, "we propose a method to transform this extended class of expressions into the commonly treated affine function class. This has a major advantage that the vast amount of loop oriented transformations techniques which are currently in use can be retained without any changes ... We believe that the same advantage is applicable in other contexts. In particular ... loop transformations are employed for ... piece-wise linear scheduling and mapping of affine dependencies").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Balasa into the system of Blickstein to have induction analysis applied to an **address expression which is piece wise linear**. The modification would have been obvious because one of ordinary skill in the art would want to use the teachings of Balasa to apply existing loop transformation techniques to gain performance advantages in situations involving piecewise linear address expressions (Balasa, p. 2, lines 10-14).

As per claim 32, the rejection of claim 31 is incorporated and further, Blickstein doesn't explicitly disclose that **the piece wise linear address expression comprise at least of one of the group comprising: a modulo operation and a division operation.**

However, Balasa, in an analogous environment, discloses that **the piece wise linear address expression comprise at least of one of the group comprising: a modulo operation and a division operation** (p. 2 line 18, "piece-wise linear" expressions are discussed, and p. 3 line 15 - 25, "this loop nest contains m indexing functions of constant modulo type ... our goal is to obtain an equivalent nest of loops ... without any modulo index").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Balasa into the system of Blickstein to have **the piece wise linear address expression comprise at least of one of the group comprising: a modulo operation and a division operation.** The modification would have been obvious because one of ordinary skill in the art would want to use the teachings of Balasa to apply existing loop transformation techniques to gain performance advantages in situations with piecewise linear address expressions, such as a modulo expression (Balasa, p. 2, lines 10-14).

As per claim 33, the rejection of claim 32 is incorporated and further Blickstein doesn't explicitly disclose that **the piece wise linear address expression comprises nested operations**.

However, Balasa, in an analogous environment discloses that **piece wise linear address expression comprises nested operations** (p. 2 lines 16-19, "loop transformations are employed for scheduling and mapping techniques of loop nests ... (and) piece-wise linear scheduling and mapping").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Balasa into the system of Blickstein to have the **piece wise linear address expression comprises nested operations.** The modification would have been obvious because one of ordinary skill in the art would want to use the teachings of Balasa to apply existing loop transformation techniques to gain performance advantages in situations involving piecewise linear, nested operations (Balasa, p. 2, lines 10-14).

As per claim 34, the rejection of claim 1 is incorporated and further, Blickstein discloses that **induction analysis is applied, wherein the induction replaces the address expression with add, accumulate, constant division and constant multiplication operations or combinations thereof** (col. 18 line 55 – col. 19 line 59, "the definition and detection of induction variables and inductive expressions will be discussed … this is a straight forward Do loop, I being the loop control variable. Notice that the inductive expression I*4 increases by 4 on each trip through the loop. By introducing a new variable I2, we can replace the multiplication with an addition, which is a less expensive operation … This optimization is known as induction variable elimination").

Blickstein doesn't explicitly disclose that at least one address expression, **is nonlinear.**

However, Balasa, in an analogous environment, discloses that at least one address expression is nonlinear (p.2 lines 5-7, "The most important practical subclass of the non-linear extensions consist of modulo expressions of affine indexing functions").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Balasa into the system of Blickstein to have induction analysis applied to at least one address expression which **is nonlinear**. The modification would have been obvious because one of ordinary skill in the art would want to utilize the teachings of Balasa to optimize nonlinear expressions because they are a very important class of expressions for the multi-dimensional signal and data processing field (Balasa, p. 2, lines 5-14).

As per claims 35-37, the Blickstein/Balasa combination also discloses such claimed limitations as addressed in claim 1, above.

7. Claims 6, 13-15, 17 and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Blickstein, U.S. Patent No. 5,577,253 in view of Balasa et al. (Balasa), "Transformation of Nested Loops with Modulo Indexing to Affine Recurrences," Parallelization Techniques for Uniform Algorithms, World Scientific Pub., 1994, p. 1-12, further in view of Janssen et al. (Janssen), "A Specification Invariant Technique for Regularity Improvement between Flow-Graph Clusters," IEEE, 1996, p. 138-143.

As per claim 6, the rejection of claim 1 is incorporated and further, Blickstein discloses that **the first, second source-level code and intermediate source-level codes generated are represented as data-flow graphs** (col. 3 lines 30-34, "Each block is also a data structure, or node, and contains pointers to its successors and predecessors ... The interlinked blocks make up a flow graph ... which is the representation used by the back end to do the optimizations").

Blickstein doesn't explicitly disclose that the first, second source-level code and intermediate source-level codes generated are represented as **shared** data-flow graphs.

However, Janssen, in an analogous environment, discloses a **shared** flow graph (p. 140, col. L, lines 28-29, "we collect the clusters in a single Shared Flow-Graph").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Janssen into the system of Blickstein to have a **shared** data-flow graph. The modification would have been obvious because one of ordinary skill in the art would want to utilize the teachings of Janssen in order to explore the search space, represented by the shared data flow graph, in an efficient way (Janssen, p. 140, col. L, lines 25-26).

As per claim 13, the rejection of claim 1 is incorporated and further, Blickstein doesn't explicitly disclose that **the method transforms the source-level codes by applying composite transformations with look-ahead local minima avoiding capabilities.**

However, Janssen, in an analogous environment, discloses that **the method transforms the** flow-graph representations of **source-level codes by applying composite transformations with look-ahead local minima avoiding capabilities** (p. 140, col. L, line 55 – col. R, line 3, composite transformations are used to change the structure of the flow-graph. Composite transformations are able to dynamically execute sequences of elementary transformations, which enables them to very dedicatedly explore a local neighbourhood and look-ahead local minima in the search trajectory").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Janssen into the system of Blickstein to have **a method** that **transforms the** flow-graph representations of **source-level codes by applying composite transformations with look-ahead local minima avoiding capabilities.** The modification would have been obvious because one of ordinary skill in the art would want to utilize the teachings of Janssen to use composite transformations that can look-ahead over undesirable regions in the search trajectory and thus avoid local minima when used in optimization (Janssen, p. 142, col. L, lines 9-13).

As per claim 14, the rejection of claim 13 is incorporated and further, Blickstein doesn't explicitly disclose that **the composite transformations comprise a finite predetermined sequence of elementary, non-decomposable transformations, wherein each of the elementary transformations are executable if a single set of conditions related to the elementary transformation are satisfied, wherein each of the composite transformations are composed such that the composite transformation can be executed if one of a plurality of sets of conditions is satisfied.**

However, Janssen, in an analogous environment, discloses that **the composite transformations comprise a finite predetermined sequence of elementary, non-decomposable transformations, wherein each of the elementary transformations are executable if a single set of conditions related to the elementary transformation are satisfied, wherein each of the composite transformations are composed such that the composite transformation can be executed if one of a plurality of sets of conditions is satisfied** (p. 140, col. L, line 55 – col. R, line 3, composite transformations are used to change the structure of the flow-graph. Composite transformations are able to dynamically execute sequences of elementary transformations (simple algebraic transformations, with very tight preconditions that check the validity of applying the transform), which enables them to very dedicatedly explore a local neighbourhood and look-ahead local minima in the search trajectory").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Janssen into the system of Blickstein to have **the composite transformations comprise a finite predetermined sequence of elementary, non-decomposable transformations, wherein each of the elementary transformations are executable if a single set of conditions related to the elementary transformation are satisfied, wherein each of the composite transformations are composed such that the composite transformation can be executed if one of a plurality of sets of conditions is satisfied.** The modification would have been obvious because one of ordinary skill art would only want to explore optimization scenarios of the flow graph that are valid and consistent with the original flow graph.

As per claim 15, the rejection of claim 14 is incorporated and further, Blickstein discloses that **the composite transformation transforms a first source-level code with a first cost into a second source-level code with a second cost, the second cost being lower than the first cost, and while executing the elementary transformation of which the composite transformation is composed of, intermediate source-level codes are generated, at least one of the intermediate source-level codes having a cost being larger than the first cost** (col. 25 lines 9-29, "A method for doing code generation … using code templates will now be described …

A template is used at different times during a compilation", and col. 25 lines 53-67, "An ILG pattern of a code generation template consists of ... a pattern tree which describes the (different) arrangement(s) of ILG(s) (i.e. representations of a second source-level code) that can be coded by this template ... An integer represents the 'cost' of the code generated by this template").

As per claim 17, the rejection of claim 15 is incorporated and further, Blickstein discloses that **the execution of the composite transformations is performed on data-flow graph representations of the source-level codes** (col. 2 lines 20-31, "The front end ... produces a representation of the program (source code) ... in the form of a ... (data-flow) graph ... after the compiler front end has generated the intermediate language (data-flow) graph and symbol table, various optimizing techniques (composite transformations) are ... implemented").

As per claim 18, the rejection of claim 17 is incorporated and further, Blickstein discloses that **applying the algebraic transformation increases the number of common subexpressions that are affected by nonlinear operations** (col. 188 lines 7-10, "said code optimization performed using ... common subexpression elimination, strength reduction, variable elimination, vectorization and loop unrolling").

8. Claims 7-9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Blickstein, U.S. Patent No. 5,577,253 in view of Balasa et al. (Balasa), "Transformation

of Nested Loops with Modulo Indexing to Affine Recurrences," Parallelization Techniques for Uniform Algorithms, World Scientific Pub., 1994, p. 1-12, in view of Janssen et al. (Janssen), "A Specification Invariant Technique for Regularity Improvement between Flow-Graph Clusters," IEEE, 1996, p. 138-143, further in view of Hong et al. (Hong), "Throughput Optimization of General Non-Linear Computations," IEEE, 1999, p. 406-409.

As per claim 7, the rejection of claim 6 is incorporated and further, Blickstein discloses that **the data-flow graphs are directed acyclic graphs** (col. 31 lines 4-5, "a CILG (data-flow graph) is a directed acyclic graph").

Blickstein doesn't explicitly disclose that the data-flow graphs are directed acyclic graphs **with homogeneous synchronous data-flow behavior**.

However, Hong, in an analogous environment, discloses that the data-flow graphs are directed acyclic graphs **with homogeneous synchronous data-flow behavior** (p. 406, col. R, line 56 – p. 407, col. L, line 2, "We use as computational model homogeneous synchronous data flow model, which is widely used in many application domains").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Hong into the system of Blickstein to have data-flow graphs that are directed acyclic graphs **with homogeneous synchronous data-flow behavior.** The modification would have been obvious because one of ordinary skill in the art would want to efficiently schedule operations by using the homogeneous synchronous data flow model (Hong, p. 407, col. L, lines 6-8).

As per claim 8, the rejection of claim 7 is incorporated and further, Blickstein doesn't explicitly disclose that **the shared data-flow graph enables modeling in a single subgraph address expressions and address calculation code with substantially different iterators**.

However, Janssen, in an analogous environment, discloses that **the shared data-flow graph enables modeling in a single subgraph address expressions and address calculation code with substantially different iterators** (p. 140, col. L, lines 28-30, "we collect the clusters in a single Shared Flow-Graph (ShFG), which is capable of representing the sharing of operations and data-dependencies between clusters").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Janssen into the system of Blickstein to have the **shared data-flow graph enable modeling in a single subgraph address expressions and address calculation code with substantially different iterators**. The modification would have been obvious because one of ordinary skill in the art would want to utilize the teachings of Janssen in order to explore a larger, global search space of optimization solutions for a program.

As per claim 9, the rejection of claim 8 is incorporated and further Blickstein doesn't explicitly disclose that **the method uses a select operation for enabling single subgraph modeling.**

However, Janssen, in an analogous environment, discloses that **the method uses a select operation for enabling single subgraph modeling** (p. 140, col. L, lines 30-32, "Expression sharing in a ShFG (shared flow-graph) is made possible by introducing a select operation, the flow graph equivalent of a hardware multiplexer").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Janssen into the system of Blickstein to have a **select operation for enabling single subgraph modeling.** The modification would have been obvious because one of ordinary skill in the art would want to make expression sharing in a shared flow-graph possible (Janssen, p. 140, col. L, lines 30-31).

9. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over Blickstein, U.S. Patent No. 5,577,253 in view of Balasa et al. (Balasa), "Transformation of Nested Loops with Modulo Indexing to Affine Recurrences," Parallelization Techniques for Uniform Algorithms, World Scientific Pub., 1994, p. 1-12, in view of Janssen et al. (Janssen), "A Specification Invariant Technique for Regularity Improvement between Flow-Graph Clusters," IEEE, 1996, p. 138-143, further in view of Miranda et al. (Miranda), "ADOPT: Efficient Hardware Address Generation in Distributed Memory Architectures", IEEE, 1996, p. 20-25.

As per claim 10, the rejection of claim 6 is incorporated and further, Blickstein doesn't explicitly disclose that **the method time-multiplexes calculations that are related to address expressions.**

However, Miranda, in an analogous environment discloses that **the method time-multiplexes calculations that are related to address expressions** (col. lines).

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of * into the system of Blickstein to have a **method** that **time-multiplexes calculations that are related to address expressions.** The modification would have been obvious because one of ordinary skill in the art would use the teachings of Miranda in order to minimize addressing and overhead costs (Miranda, p. 1 col. L lines 18-19 and p. 1 col. R lines 21-25).

10. Claims 24 - 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Blickstein, U.S. Patent No. 5,577,253 in view of Balasa et al. (Balasa), "Transformation of Nested Loops with Modulo Indexing to Affine Recurrences," Parallelization Techniques for Uniform Algorithms, World Scientific Pub., 1994, p. 1-12, further in view of Kathail et al. (Kathail), U.S. Patent No. 5,692,169.

As per claim 26, the rejection of claim 25 is incorporated and further Blickstein discloses that **code hoisting is considered for applying** to **expressions with overlapping ranges** (col. 3 lines 49-52, "This mechanism is used by the global optimizer to determine (using range analysis) legal and effective optimizations, including ... code motions").

As per claim 27, the rejection of claim 25 is incorporated and further Blickstein discloses that **code hoisting is applied to expressions with at least one common factor** (col. 2 lines 30-31, "various optimizing techniques are used", and col. 3 lines 50-52, "including common subexpression expression recognition and code motions").

As per claim 28, the rejection of claim 25 is incorporated and further Blickstein discloses that **code hoisting is applied for equal expressions when the sum of their ranges is larger than the overall range** (col. 3 lines 49-52, "This mechanism is used by the global optimizer to determine (using range analysis) legal and effective optimizations, including ... code motions").

As per claim 29, the rejection of claim 25 is incorporated and further Blickstein discloses that **code hoisting is applied for non-equal expressions after a cost-benefit analysis that evaluates: the degree of similarity of the non-equal expressions, the degree of similarity being expressed as the amount of common subexpressions within the non-equal expressions, their costs and the cost after a**

As per claim 24, the rejection of claim 21 is incorporated and further Blickstein doesn't explicitly disclose that **code hoisting is applied across the scope of a conditional**.

However, Kathail, in an analogous environment, discloses that **code hoisting is applied across the scope of a conditional** (col. 2 lines 9-13, "To optimize a program, code may be moved above a conditional branch in a scheduling process called speculative code motion. Speculative code motion refers to the movement of an instruction above a conditional branch that controls its execution").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Kathail into the system of Blickstein to have code hoisting **applied across the scope of a conditional**. The modification would have been obvious because one of ordinary skill in the art would wand to use code hoisting applied across the scope of a conditional in order to enhance instruction level parallelism (Kathail col. 2 lines 17-18).

As per claim 25, the rejection of claim 24 is incorporated and further Blickstein discloses that **code hoisting is based on a range analysis** (col. 3 lines 49-52, "This mechanism is used by the global optimizer to determine legal and effective optimizations, including ... code motions", and a code hoisting optimization scheme must use range analysis in order to local legal and effective code hoisting opportunities).

potential code hoisting step (col. 2 lines 30-31, "various optimizing techniques are used", and col. 3 lines 49-52, "This mechanism is used by the global optimizer to determine legal and effective optimizations, including common subexpression expression recognition and code motions").

Conclusion

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Andre R. Fowlkes whose telephone number is (703)305-8889. The examiner can normally be reached on Monday - Friday, 8:00am-4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (703)305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

[illegible]

ARF

WEI Y. ZHEN
PRIMARY EXAMINER